Express Mail No. EL280661986US

PATENT APPLICATION
DOCKET NO.: SNS-006 (7268/10)

SYSTEMS AND METHODS FOR INTERACTING WITH VIRTUAL OBJECTS IN A HAPTIC VIRTUAL REALITY ENVIRONMENT

CROSS –REFERENCE TO RELATED APPLICATION

This application is based on and claims priority to U.S. Provisional Patent Application Serial No. 60/093,304, filed July 17, 1998, entitled "Method and Apparatus for Sculpting Virtual Objects in a Haptic Virtual Reality Environment," the entire contents of which are incorporated herein by reference.

FIELD OF INVENTION

The invention relates generally to a method and apparatus for interacting with virtual objects in a haptic virtual reality environment, and more specifically to a method and apparatus for using virtual tools to interact with virtual objects in a haptic virtual reality environment.

BACKGROUND OF INVENTION

Computers have been long used to display objects in a computerized environment. Examples are CAD (computer-aided design) systems that are used in the design of mechanical parts, such as parts used in an assembly process. For example, a designer may use a CAD system to design parts for an automobile, which are then produced and used in an automobile assembly line. Such CAD systems require significant training and understanding in using the computerized design system. CAD systems are typically difficult to use and lack the freedom

and expressiveness of traditional noncomputerized drawing, sketching, and model-making procedures.

A more modern approach uses a virtual reality technique to model objects in a computerized virtual environment. Virtual Reality (VR) is an artificial environment constructed by a computer which permits the user to interact with that environment as if the user were actually immersed in the environment. Early VR devices permitted the user to see three-dimensional (3-D) depictions of an artificial environment and to move within that environment. The reality of the VR environment is enhanced by the ability of a user to manipulate virtual objects within the virtual environment using hand motions and gestures. A designer may use a virtual tool to manipulate and/or modify a computerized model or virtual object in the virtual environment. However, a designer must be able to accurately evaluate the virtual object. One way the designer can evaluate the virtual object is by touching and feeling the surface of the object by using a virtual tool which the designer controls through a haptic interface device, such as a joystick, glove, stylus or other physical device.

Many existing virtual reality techniques do not provide for a realistic feeling of interaction by the designer. The designer cannot feel when virtual tools touch the virtual object. Moreover, in some cases, the virtual tool may pass through the virtual object without any impediment, thereby severely degrading the realism of the experience and the designer's ability to feel all parts of the virtual object accurately.

SUMMARY OF INVENTION

Thus, there is a need for a 3-D computerized modeling system that overcomes the problems of CAD techniques and traditional noncomputerized

modeling techniques. One object of the invention is to provide a haptic feedback approach that aids the user of the system in evaluating the whole object including special features by giving the user useful interactive force feedback. This feedback is particularly useful when the user skirts the surface of a virtual object and encounters a special case such as an edge, a "hole," or other feature in the virtual object, or attempts to penetrate it.

Being able to feel the virtual object allows the user to resolve visual ambiguities, such as a shape that may appear either concave or convex as perceived by an observer. The user may rely on haptic feedback when modifying the object such as scratching a slight groove in the object, which the user then deepens or expands while receiving feedback through the tool on the shape and current depth of the groove. Feedback also allows the designer to monitor and modulate the cutting rate or amount of change induced by the virtual tool. Haptic feedback also helps the user navigate around and on the surface of the object; that is, using the feel of the object to know where the virtual tool is on the object.

The system of the invention may be used for different purposes such as in a medical teaching system, in other educational systems, for seismic data analysis and interpretation, in a computer game, and other purposes.

The invention relates to a method for interfacing with a virtual object in a haptic virtual environment, including the steps of generating a virtual object including a virtual surface in the haptic virtual environment; sensing a location of a user in real space; determining a virtual tool having discrete points for use by the user in the haptic virtual environment; determining a haptic interface location in the haptic virtual environment in response to the location of the user in real space; determining locations for the points of the virtual tool in the haptic virtual environment in comparison to the haptic interface location and the location of the

virtual surface; determining a geometry for the virtual surface at an area of penetration of the virtual tool if one or more of the points penetrates the virtual surface; and limiting movement of the virtual tool based on (i) the geometry of the virtual surface, (ii) the location of one or more points, and (iii) the haptic interface location.

In another embodiment of the invention, the method includes limiting movement of the virtual tool by moving the position of the virtual tool toward the haptic interface location. In a further embodiment, the method includes determining a surface direction vector in response to the location of one or more points of the virtual tool and determining if one or more of the points penetrates the virtual surface. In another embodiment, the method includes determining the geometry of the virtual surface to be an edge geometry. In another embodiment, the method includes calculating an interaction force between the virtual object and virtual tool in response to determining the locations of the points of the virtual tool.

The invention also relates to a system for interfacing with a virtual object in a haptic virtual environment. The virtual object includes a virtual surface. The system also includes a haptic interface device, a virtual tool, and a modeling application. The haptic interface device senses a location of a user in real space. The virtual tool includes a plurality of discrete points for use by the user in the haptic virtual environment. The modeling application is in communication with the haptic interface device, the virtual object, and the virtual tool, and the modeling application (a) determines a haptic interface location in the haptic virtual environment in response to the location of the user in real space; (b) determines locations for the points of the virtual tool in the haptic virtual environment in comparison to the haptic interface location and the location of the

virtual surface; (c) determines a geometry for the virtual surface at an area where one or more points of the virtual tool penetrates the virtual surface; and (d) limits movement of the virtual tool based on (i) the geometry of the virtual surface, (ii) the location of one or more points of the virtual tool, and (iii) the haptic interface location.

In one embodiment, the modeling application limits the movement of the virtual tool by moving the virtual tool toward the haptic interface location. In another embodiment, the system includes a surface direction vector that the modeling application determines based on the locations of one or more points of the virtual tool that have penetrated the virtual surface. In another embodiment, the geometry of the virtual surface is an edge geometry. In a further embodiment, the modeling application calculates an interaction force between the virtual object and the virtual tool based on the locations of the points of the virtual tool.

In another embodiment, the invention relates to a method for interfacing with a virtual surface in a haptic virtual environment, including the steps of generating a virtual surface in the haptic virtual environment; sensing a location of a user in real space; determining a virtual representation of the user in real space, the virtual representation comprising a plurality of discrete points; determining a haptic interface location in the haptic virtual environment in response to the location of the user in real space; determining a virtual representation location in the haptic virtual environment; moving the virtual representation location toward the haptic interface location in the haptic virtual environment; and limiting movement of the virtual representation based on a geometry of the surface and on preventing any one of the plurality of discrete points of the virtual representation from substantially penetrating the virtual surface.

BRIEF DESCRIPTIONS OF THE DRAWINGS

The invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1 depicts a functional block diagram of a system for one embodiment of the invention, including a haptic interface device, modeling application, and graphics display;

FIG. 2A provides a pictorial view of a virtual environment including a virtual object and a virtual tool, for one embodiment of the invention;

FIG. 2B provides a pictorial view of a virtual tool contacting the virtual surface of a virtual object in connection with a haptic interface location within the virtual object, for the embodiment of the invention shown in FIG. 2A;

FIG. 3 illustrates a high-level flowchart of the haptic rendering process between a virtual tool and virtual object, for one embodiment of the invention;

FIG. 4 is a high-level flowchart of the haptic rendering process between a virtual tool and a virtual object for another embodiment of the invention;

FIG. 5A illustrates a pictorial view of a virtual tool approaching the virtual surface of a virtual object;

FIG. 5B illustrates a pictorial view of a proposed location for a virtual tool with one point of the virtual tool penetrating the virtual surface of the virtual object, for the embodiment shown in FIG. 5A;

FIG. 5C illustrates a pictorial view of a proposed location for the virtual tool with two points of the virtual tool penetrating the virtual surface of the virtual object, for the embodiment shown in FIG. 5A;

FIG. 5D illustrates a pictorial view of a virtual tool moving along the virtual surface of a virtual object, for the embodiment shown in FIG. 5A;

FIG. 5E illustrates a two dimensional pictorial view of a virtual tool moving along a concave edge in the virtual surface of a virtual object for the embodiment shown in FIG. 5A;

FIG. 5F illustrates a three dimensional pictorial view of a virtual object moving along a concave edge, for the embodiment shown in FIG. 5E;

FIG. 6 illustrates a virtual tool with points located throughout the interior of the tool;

FIGS. 7A-7C depict flowcharts of the haptic rendering process between a virtual object and virtual tool for one embodiment of the invention;

FIG. 8A illustrates a pictorial view of a virtual tool encountering the convex edge of a virtual object for one embodiment of the invention;

FIG. 8B illustrates a pictorial view of a surface direction vector calculated for one proposed tool position for the embodiment of the invention shown in FIG. 8A;

FIG. 8C illustrates a pictorial view of a surface direction vector calculated for a second proposed tool position for the embodiment shown in FIG. 8A;

FIG. 8D illustrates a pictorial view of virtual tool constrained to the edge of a virtual object for the embodiment of the invention shown in FIG. 8A;

FIG. 9 depicts a set of voxels with different density values and an isosurface;

FIG. 10 illustrates a two dimensional representation of a rectangular solid with voxels and an isosurface, for the embodiment shown in FIG. 9;

FIG. 11 illustrates a ramp length diagram comparing voxel values to penetration distances for one embodiment of the invention;

FIG. 12 illustrates a pictorial view of the surface direction vector of a point and surrounding density evaluation points for one embodiment;

FIG. 13 illustrates a schematic view of a gradient for a virtual object, including a last SCP, a current SCP, an initial point, and a midpoint for one embodiment;

FIGS. 14A, 14B, and 14C illustrates pictorial views of a virtual surface and the endpoints of a segment that intersects the a virtual surface of a virtual object for one embodiment of the invention;

FIG. 15 illustrates a pictorial view of previous surface contact points, tangency planes, and resulting surface contact point for one embodiment of the invention;

FIGS. 16A and 16B show a pictorial view of a spherical virtual tool in a channel formed between two virtual objects and;

FIGS. 17A-17E show pictorial views of a virtual tool encountering a surface and moving along the surface constrained by a constraint plane.

DETAILED DESCRIPTION OF THE INVENTION

The description includes headings and subheadings that aid in organizing the text, but are not meant to be limiting in any way. Topics discussed under a

heading or subheading may also be described elsewhere throughout the specification.

FIG. 1 shows a system for one embodiment of the invention, including a haptic interface device 10, modeling application 12, and graphics display 14 in electrical communication with each other. The modeling application 12 includes a haptic rendering process 16, a virtual tool and object interaction process 18, a virtual object modification process 20, and a graphics process 22 all in electrical communication with each other.

Generally, a user of the system uses the haptic interface device 10 to interact with the virtual object 26 (see FIG. 2) receiving force feedback produced by the haptic rendering process 16 and viewing graphics rendered by the graphics process 22 on a graphic display 14.

Software Process

In one embodiment, a "process", as referred to in FIG. 1, is a software process executing on a hardware microprocessor. All the processes may execute on one microprocessor, or in other embodiments, one or more processes may execute on different microprocessors, which are linked by buses, cables, local networks, wide area networks, or global computer networks, such as the Internet.

The modeling application 12 is viewed in FIG. 1 as a software application executing on one computer system. In another embodiment, the modeling application 12 executes on one or more computer systems connected by a communications device, such as a bus, cable, or network connection. In an alternate embodiment, the modeling application is a hardware device, such as an ASIC (application specific integrated circuit), and one or more processes of the application are implemented on one or more ASIC devices. In a further

embodiment, the modeling application 12 is implemented as one or more objects, which may execute on one or more computer systems. In one embodiment, the modeling application 12 runs on a dual 300 MHz Intel® Pentium® 2 computer running Microsoft® Windows NT™ 4.0 using an Open GL accelerated graphics card.

The modeling application 12 is not required to include a haptic rendering process 16, an interaction process 18, a modification process 20, and a graphics process 22. In one embodiment, the functions of the modeling application 12 are implemented by a different number of processes. In one embodiment, the modeling application 12 includes the haptic rendering process 16 and the graphics process 22.

In one embodiment, the invention is implemented using an object-oriented approach. The haptic rendering process 16 and other processes are implemented as software objects. In another embodiment, the virtual object 26 and the virtual tool 28 are implemented as software objects and perform one or more of the functions of the haptic rendering process 16. In one embodiment, the virtual tool 28 is a software object that performs such functions as determining if contact has occurred with a virtual object 26 and determining the surface direction vector 101, as will be discussed later.

In one embodiment, the virtual object 26 and the virtual tool 28 are implemented as software objects in the C++ programming language. In other embodiments, the virtual object 26 and virtual tool 28 are implemented using an object-oriented programming language other than C++.

In one embodiment, the modeling application is a computer program stored on a computer readable storage media, such as a CD disc, diskette, tape, or other media. In another embodiment, the modeling application is a computer

program distributed over a computer-readable propagated signal, such as a program distributed over the Internet.

Haptic Interface Device

In one embodiment, the system includes a haptic interface system, as shown in FIG. 1, including the haptic interface device 10 and the haptic rendering process 16 which generates a virtual object of the virtual environment to be "touched" and determines the results of the interaction (discussed in more detail below). The haptic interface device 10 is a tactile or force-feedback device which provides the touch sensations of interacting with virtual objects 26 to a user of the system. Some haptic interface devices 10 consist of an electro-mechanical linkage which can exert a controllable force on a user's hand. See, for example, U.S. Patent No. 5,625,576 issued to Thomas H. Massie and J. Kenneth Salisbury, Jr., the disclosure of which is herein incorporated by reference in its entirety. As used herein, "haptic rendering" refers to the creation of a virtual environment with which a user can interact through the sense of touch. The term "haptic rendering process" 16 refers to the computer program which generates the haptic aspects of the virtual environment and determines the forces to be applied to a user through a haptic interface. The haptic rendering process 16 generates haptic representations of virtual objects in the virtual environment.

Overview of Device, Virtual Object and User Interaction

FIG. 2A shows a haptic virtual environment including a virtual object 26 and a virtual tool 28. The virtual object 26 of the embodiment shown in FIG. 1 is depicted as a 3-D (three dimensional) block of material typically "floating" in the virtual space of the virtual environment. The virtual object 26 has a virtual surface

25 that represents the "skin" of the virtual object 26. The virtual tool 28 is represented in FIG. 2A as a sphere 34 with a rod or "handle" 32 connected to it.

In one embodiment, the user uses a haptic interface device 10 in real space to grasp or manipulate the handle 32 of the virtual tool 28 in virtual space. In one embodiment, the location of this handle with respect to the virtual tool 28 can be changed interactively by the user. As used herein, a "haptic virtual environment" refers to a computer-generated virtual environment that can be explored by a user through the sense of touch. In one embodiment, the haptic virtual environment contains a virtual object 26 that is model of a real world object that a user is creating in the virtual environment. In another embodiment, the haptic virtual environment incorporates two or more virtual objects 26 that are linked to each other, such as in a hierarchical arrangement. It should be understood that the interaction and/or modification methods described herein may be readily extended to apply to two or more virtual objects 26 linked or associated in a haptic virtual environment.

FIG. 2B illustrates a virtual tool 28 contacting the virtual surface 25 of a virtual object 26. The user guides the virtual tool 28 using the haptic interface device 10, represented, in this embodiment, by a stylus 33 in Fig. 2B. The position and orientation of the tip of the stylus 33 indicate the haptic interface location 98. Note that, although the user may be manipulating a literal stylus in some embodiments, the haptic interface location 98 could be controlled by a user interacting with any number of differently shaped elements such as a thimble, a yoke, or a ball. The tip of the virtual stylus 33 is indicated by the haptic interface location 98. In one embodiment, the haptic rendering process 16 tracks the haptic interface location 98, but does not otherwise track the shape or location of the entire haptic interface device 10.

The haptic rendering process 16 attempts to move the virtual tool 28 so that the origin 27 of the virtual tool 28 matches the haptic interface location 98. However, unless the haptic rendering process 16 is using the virtual tool 28 to remove material from the virtual object 26, then the haptic rendering process 16 typically does not allow the virtual tool 28 to penetrate the virtual object 26. Thus, as shown in FIG. 2B, the user has attempted to move the virtual tool 28 into the virtual object 26, which is indicated by the haptic interface location 98 within the virtual object 26. The haptic rendering process 16 calculates a resistance to the movement of the virtual tool 28 into the virtual object 26. This calculation is based on a connection 29 between the tool origin 27 and the haptic interface location 98, as will be discussed in more detail later. In one embodiment, the connection 29 includes a virtual spring 31. In one embodiment, the connection 29 includes a virtual dash-pot. Thus, if the user attempts to move the virtual tool 28 further into the virtual object 26, the haptic rendering process 16 calculates an increasing resistance force that is fed back to the user through the haptic interface device 10 based on the virtual spring 31.

In one embodiment, the user is allowed to move the virtual tool 28 through the virtual object 26 without resistance while removing material. In this case, the user selects a transparent or translucent mode, and the virtual tool 28 appears translucent. The haptic rendering process 16 allows the user to move the virtual tool 28 through the virtual object 26 without constraint or resistance.

Description of Virtual Tool and Modification Options

As already described, the user interacts with the virtual object 26 in the virtual environment through a virtual tool 28. The user may select any shape for the tool 28. The shape of the tool 28, along with other characteristics, such as interaction mode, determine the interaction with the virtual object 26. In one

embodiment, the tool 28 may be represented as a series of discrete points in virtual space which outline a three-dimensional shape of the tool 28. The virtual tool 28 is modeled as a set of discrete points for the purposes of haptic interaction and collision detection with the virtual object 26. In another embodiment, the points of the virtual tool 28 are created by an algebraic equation or any other continuous or piecewise mathematical method suitable for determining a 3-D shape in a virtual environment. In another embodiment, the tool 28 can be represented directly by continuous or piecewise mathematical equations, rather than by discrete points. The virtual tool 28 may take on any of a number of shapes that may be useful for a user when using a virtual tool 28 to create a virtual object 26 in the virtual environment. Typical shapes may include a sphere or cylinder. In another embodiment, the user selects one or more interaction modes for the virtual tool 28, such as a sandpaper mode, which causes the tool 28 to remove material gradually from the virtual object 26, much like using real sandpaper to smooth the shape of a block of wood in the real world.

Creating and Manipulating the Virtual Object

FIG. 3 illustrates a flowchart of the haptic rendering process between a virtual tool 28 and virtual object 26. First, a virtual object 26 is generated (step 40). Typically, this occurs when the user requests that an initial virtual object 26 be created, for example, by directing that the virtual object 26 assume a 3-D cube or "block" shape. In one embodiment, the initial virtual object can be defined as a shape generated from a saved file, scanner, or 3D digitizer. In one embodiment, the user selects an interaction mode that selects the characteristic of the virtual object 26. The shape and material, and surface properties of the virtual object 26 can be specified, for example, the hardness or softness of the object 26. For example, if the user selects a sculpting mode, then the virtual object 26 assumes

characteristics generally representative of a block of clay. If the user selects a 3-D sketch mode, then the virtual object 26 is overlaid with a stack of planes or slices, and the user sketches a template on the surface of one of the planes. However, the virtual object 26 is not required to assume a cube or block shape and may assume any 3-D shape that the user finds useful when starting to design a model from the virtual object 26. In one embodiment, the object properties permit only manipulation or movement of the virtual object 26 without any sculpting or other modification thereof. The modification mode is not limited to what is described here, but, in other embodiments, is based on other modes of interaction or modification that a user finds useful. For example, other modes may include smoothing the surface of the virtual object 26, shifting material, or mirroring all or parts of the object 26.

In one embodiment, the virtual object 26 is created by the modification process 20 under the directions of the user, and then the graphics process 22 displays a corresponding representation of the virtual object 26 to the user.

If the user selects a 3-D sketch mode, then the virtual object 26 is overlaid with a stack of planes or slices, and the user sketches a template on the surface of one of the planes.

Generating the Virtual Tool

In the next step, a virtual tool 28 is generated (step 42). In one embodiment, the user selects a shape and characteristics for the virtual tool 28, which the haptic rendering process 16 generates in the virtual environment. The graphic process 22 then displays a corresponding version of the virtual tool 28 to the user on a graphics display 14. For example, the virtual tool 28 can assume different shapes and interaction or modification characteristics. The tool 28 can

assume any 3-D shape, such as a sphere 34 or a cube, or a substantially 2-D shape, such as a spatula or knife. In general, modes include a material removal, material addition, or other material modification mode such as smoothing. In one embodiment, removal modes include Boolean removal, sand paper removal, and removal using the concept of a function which falls off gradually. Additional modes include Boolean addition, Boolean pull, and pull using a fall-off function. An additional form of interaction and modification includes a sketch mode for sketching a template on the surface of a virtual object 26. Used here, Boolean refers to adding or subtracting two geometries to arrive at a third geometry.

Sensing User Location

In the next step, sensors determine the location of a user in real space (step 44). In one embodiment, the sensors are any type of sensors useful in determining the location of a user, such as the location of a hand in real space. Such sensors could be based on measurements of location based on mechanical, electrical, magnetic, optical, or other sensing devices. In one embodiment, the haptic interface device 10 senses the location of the user in real space. For example, the user physically manipulates the haptic interface device 10, such as a handgrip or stylus, in real space and the location of this device is determined in real space. In one embodiment, one such haptic interface device 10 is the PHANToM® device from SensAble Technologies, Inc., Cambridge, Massachusetts. Generally, the PHANToM® device can sense six degrees of freedom – x, y, z, pitch, roll, yaw, while providing for force feedback in three degrees of freedom – x, y, z. One embodiment of this invention includes a haptic interface that can provide more than three degrees of force feedback.

As used herein, "real world space" is defined as the real world environment. The haptic rendering process 16 utilizes the information obtained

by the sensors to determine the haptic interface location 98 in the virtual environment. As used herein, "haptic virtual environment" refers to the region in the computer generated virtual environment with which the user can interact through the sense of touch. The location of the haptic interface describes the location of the user in the haptic virtual environment.

Correlating User Location and Virtual Tool Position

In one embodiment, the haptic rendering process 16 then translates the location of the haptic interface device 10 in real space into a corresponding location in the haptic virtual environment, which is the haptic interface location 98 (step 46). Then the haptic rendering process 16 uses a method to limit the movement of the tool 28 based on the virtual surface 25, the position of the tool 28, and the haptic interface location 98 (step 54). The objective of the method is to move the tool 28 as close as possible to the haptic interface location 98, without crossing the virtual surface 25 and using a path for the tool 28 that yields progressively better locations for the tool 28, which typically means locations closer to the haptic interface location 98. This method is discussed in detail later.

Typically, the haptic rendering process 16 analyzes the interaction and calculates results based on the potential position of the virtual tool 28 relative to the virtual object 26 (step 56). Then the results are applied to the virtual tool 28 and/or the user (step 58).

Calculation of Force

For example, if the movement of the virtual tool 28 is constrained such that the virtual tool origin 27 is not coincident with the haptic interface, the haptic rendering process 16 may calculate an interaction force to be applied to the haptic interface device 10, so that the user feels a resistance to trying to move the virtual

tool 28 into the virtual object 26. In this case, the results are a feedback force applied to the user via the haptic interface device 10 and corresponding constraints or limits on the movement of the virtual tool 28 (step 56). The force feedback provides the user important non-visual information about the shape of the virtual object, whether the interaction mode is object modification or simply evaluating the shape of the object.

Interaction and Modification

In one embodiment, if the user is using the virtual tool 28 in an interaction mode, such as sandpaper, then the modification process 20 calculates changes in the virtual object 26, such as material being removed, which in turn changes the graphical representation of the virtual object 26. The results, in this case, are a modification to the virtual object 26 (step 58). Interaction modes need not result in modification of the virtual surface. For example, in another case, the user may be trying to use the virtual tool 28 to evaluate the shape of the virtual object 26 without trying to modify the object 26. In this case, the results are limits on the movement of the virtual tool 28 without any penetration or modification of the virtual object 26. Another example is if the user presses the virtual tool 28 into a virtual object 26 in an erase or removal mode, but does not press with enough force, then the virtual tool 28 remains at the surface or may skirt along the surface of the virtual object 26 without removing any material. The results, in this case, are constraints on the movement of the virtual tool 28 (step 58). The calculation and application of results (steps 56 and 58) are not confined to what is described here but involve other effects depending on many factors, such as the shape of the virtual tool 28, the characteristics of the tool 28, the characteristics of the virtual object 26, the nature of the movement of the virtual tool 28 relative to the virtual object 26, and other factors. Another factor may be a construction constraint that

aids in the construction of a virtual object 26. In one embodiment, the constraint can be a line, an arbitrary curve, or a surface that constrains the movement of the virtual tool 28.

Ongoing Interaction between User and System

Finally, after the results have been applied (step 58), the user engages in additional movement of the haptic interface device 10, in which case a new location must be sensed (step 44) and steps 46-58 are repeated. Alternatively, the user changes the nature of the virtual object 26 and/or the interaction mode. The user may also change the shape or characteristics of the virtual tool 28 (not shown in FIG. 3). These changes by the user in turn would affect the calculations, constraints, and results determined by steps 46, 54, 56, and 58.

VIRTUAL OBJECT AND TOOL INTERACTION

Haptic Rendering Process

FIG. 4 is a flowchart of the haptic rendering process 16 for the interaction between a virtual object 26 and a virtual tool 28 for one embodiment of the invention. In the first steps, the haptic rendering process 16 generates a virtual object 26 (step 60). The haptic rendering process 16 determines or generates a virtual tool 28 represented using a plurality of discrete points for use by the user (step 62). Sensors sense the location of the user in space (step 64) in a manner similar to that described for FIG. 3 above. The haptic rendering process 16 then determines a haptic interface location 98 (see FIG. 5A) for the haptic interface (step 66) in the haptic virtual environment corresponding to the location of the haptic interface device 10, which the user is manipulating in real space. The haptic rendering process 16 determines potential locations for the points of the

virtual tool 28 in the haptic virtual environment in comparison to the haptic interface location 98 and the virtual surface of the virtual object 26 (step 68).

The haptic rendering process 16 determines the amount of penetration into the virtual object 26 for all the points of the virtual tool 28 if it were to be moved to the potential location (step 70). The haptic rendering process 16 may determine that there would be no penetration, that only one point of the virtual tool 28 would penetrate the virtual object 26, or that several points of the virtual tool 28 would penetrate the virtual object 26.

If at least one of the points of the virtual tool 28 has penetrated the virtual object 26, then the haptic rendering process 16 determines a geometry for the virtual surface at the area of penetration of the virtual tool 28 (step 72). For example, the haptic rendering process 16 determines if the virtual object 26 has an edge, trough, valley, vertex, or hole in the vicinity of the virtual tool 28, to be discussed in more detail later with respect to FIGS. 7A-7C. This determination is then used in the next step (step 74), which determines limits or constraints for the movement of the virtual tool 28 based on the geometry of the virtual object 26 (as determined in step 72), the locations of the points of the virtual tool 28 that would have penetrated the virtual object 26 (as determined in steps 68 and 70), and the haptic interface location 98 (as determined in step 66). The haptic rendering process 16 then uses the previously determined constraints to constrain or limit the movement of the virtual tool 28 (step 76). For example, if the virtual tool 28 has encountered an edge or trough on the surface of the virtual object 26, then the virtual tool 28 may be constrained to slide along the edge or trough (see FIG. 5E) until the user makes a definitive movement, such as moving away from the virtual object 26. In one embodiment, the movement of the virtual tool 28 is limited without the user feeling any force feedback. In another embodiment, the

movement of the tool 28 is limited and the user feels an interactive force feedback corresponding to the limits on movement of the tool 28.

Relationship of Virtual Tool Location to Haptic Interface Location

FIG. 5A illustrates a virtual tool 28 approaching the virtual surface 25 of a virtual object 26. The surface of the virtual tool 28 is defined by a series of points, S1, S2, S3, S4, S5, S6 in a simplified view of a virtual tool 28 in one embodiment.

In another embodiment of the invention, the points that define the volume of the virtual tool 28 extend throughout the space of the virtual tool 28 (shown as additional interior points 112 within the tool 28 in FIG. 6*)*. In one embodiment, the additional points 112 are spaced evenly throughout the interior 3-D volume of the tool 28. In another embodiment, there is no requirement that the additional points 112 be spaced evenly within the tool 28.

In the example shown in FIG. 5A, the user is moving the haptic interface device 10 so that the virtual tool 28 is moving toward the virtual surface 25 of the virtual object 26. The haptic rendering process 16 attempts to move the origin 27 of the virtual tool 28 to match the haptic interface location 98. In the embodiment shown in FIG. 5A, the location of the virtual tool origin 27 lags behind the haptic interface location 98 as the user moves the virtual tool 28 through the virtual space.

Haptic Rendering and Graphics Rendering

It should be understood that the haptic rendering process 16 is operating at a high rate of speed, such as updating the location of the haptic interface location 98 and virtual tool 28 many times per second, as the user attempts to move the virtual tool 28 through the virtual environment. In one embodiment, the haptic

rendering process 16 is updating the locations at about 1000 times per second. In one embodiment, some calculations of the haptic rendering process, such as force reaction calculations, are occurring at about 1000 times per second, while less time critical calculations of the haptic rendering process, such as geometry calculations, are occurring at slower rates, such as 100 times per second. The graphics process 22 updates the image seen by the user on the graphics display 14, but typically at a lower refresh rate than the haptic rendering process 16. In one embodiment, the graphics process 22 updates the graphics display 14 at a rate of about 60 times per second.

Example of Tool Points Penetrating Virtual Object

FIG. 5B illustrates one point S4 of a virtual tool 28 encountering the virtual surface 25 of the virtual object 26, for the embodiment shown in FIG. 5A. The haptic rendering process 16 attempts to move the virtual tool origin 27 to match the location of the haptic interface 98, but determines that a point, S4, of the virtual tool 28 would cross the virtual surface 25. The haptic rendering process 16 then determines the approximate minimum distance vector 101 in a direction toward the virtual surface for a point S4. This vector 101, later referred to as the surface direction vector 101, also has the property that it is the approximate normal vector to the virtual surface 25 where it crosses the surface 25. As described in more detail later, the haptic rendering process 16 uses this vector 101 to calculate movement constraints for the virtual tool 28, as the virtual tool origin 27 is progressively moved toward the haptic interface location 98. In general, the haptic rendering process 16 attempts to keep the points of the tool 28 outside of the virtual surface 25 of the virtual object 26; that is, the object 26 is treated as a solid.

Example of More than One Point Encountering the Virtual Object

FIG. 5C illustrates two points, S3 and S4, of the virtual tool 28 encountering the virtual surface 25 of a virtual object 26, for the embodiment shown in FIG. 5A. In this case, the virtual tool 28 encounters the virtual surface 25 at a different orientation than the one shown in FIG. 5B. For example, the user has rotated the virtual tool 28 slightly or is moving the virtual tool 28 at a slightly different angle, so that two points, S3 and S4, are encountering the virtual surface 25 rather than the one point, S4, as shown in FIG. 5B. The surface direction vector 101 shown in FIG. 5B is based on point S4. In FIG. 5C, the surface direction vector 101 is based on point S4 as it has penetrated the virtual surface 25 more than point S3. In the embodiment shown, if multiple points penetrate the virtual surface 25, the surface direction vector 101 is calculated for the point of deepest penetration. Other embodiments may use a combination of all points which are penetrating to calculate a resulting minimum surface direction vector 101.

Local Geometry Constrains Motion

FIG. 5D illustrates a virtual tool 28 skirting along the virtual surface 25 of a virtual object 26, for the embodiment shown in FIG. 5A. A plane of tangency or constraint plane 104 is formed that is orthogonal to the surface direction vector 101 that was previously calculated. In FIG. 5D, the plane of tangency 104 is shown aligned with the origin 27 of the virtual tool 28. In other embodiments, the plane of tangency 104 may be aligned differently, such as at or near the virtual surface 25 of the virtual object 26, as long as the plane of tangency 104 constrains the movement of the virtual tool 28 substantially parallel to the virtual surface 25. In FIG. 5D, the plane of tangency extends out of the plane of the diagram. The virtual tool 28 is allowed to move along the plane of tangency 104. The

movement is shown by the direction of movement or motion vector 106a.

FIG. 5E provides a two dimensional view of a virtual tool 28 constrained to move along an inside or concave edge 108 . FIG. 5F illustrates a l three dimensional view of the virtual tool 28 constrained to move along an edge 108 of a trough or valley 110 for the embodiment of the invention shown in FIG. 5E. The virtual tool 28 is constrained to move in the direction of a vector 106b that constrains the tool 28 to move in a direction approximately parallel to the concave edge 108 toward a haptic interface location 98. FIG. 5F provides another pictorial view of a the tool 28 moving along a vector 106b approximately parallel to the concave edge 108 of the virtual object 26.

Modification of the Virtual Object

If the virtual tool 28 is in a removal mode, the modification process 20 may also determine material to be removed from the virtual object 26 in response to the shape of the virtual tool 28. In one embodiment, this removal mode is termed a carving mode. In one embodiment, the user feels a resistive force when attempting to move the virtual tool 28 into the virtual object 26 because forward motion of the virtual tool 28 into the object 26 is not allowed. Forward motion of the tool 28 is only achieved as a result of the modification process 20 changing or deforming the geometry of the virtual object 26 in a manner that simulates carving of the virtual object 26. The haptic rendering process 16 continues to treat the virtual object 26 as a solid object as material is removed or added. For example, the user feels added material as solid, since movement of the virtual tool 28 into the material is not permitted.

Case 1 of Tool in Empty Space

FIGS. 7A-7C depicts flowcharts of the haptic rendering process 16 which calculates interactions between the virtual object 26 and the virtual tool 28 for one

embodiment of the invention. These flowcharts represent movement of the virtual tool 28 in steps relative to the virtual surface 25 of a virtual object 26. First, the haptic rendering process 16 proposes to move the virtual tool origin 27 toward the haptic interface location 98 (step 120). Then the haptic rendering process 16 determines if any of the tool 28 points would penetrate the virtual surface 25 (step 122). If none of the tool 28 points would penetrate the virtual surface 25, this represents "Case 1." The virtual tool 28 would encounter only empty space by moving to the proposed new location, so the incremental movement of the virtual tool origin 27 is allowed (step 121) and the haptic rendering process 16 returns to step 120.

Case 2 of Tool Penetrating the Virtual Surface

If at the proposed tool position, some points of the virtual tool 28 would penetrate the virtual surface 25 (step 122), then the haptic rendering process 16 finds the direction of shortest distance to the virtual surface as indicated by surface direction vector 101 for the point of the greatest potential penetration into the virtual object 26 by the virtual tool 28 (step 126, see also FIGS. 5B and 5C). The haptic rendering process 16 then calculates a constraint plane or plane of tangency 104 (see FIG. 5D) based on the surface direction vector 101 (step 127 of FIG. 7A). The plane of tangency 104 is a plane orthogonal to the surface direction vector 101. The haptic rendering process 16 then attempts to move the virtual tool origin 27 toward the haptic interface location 98 but constrains the movement to the plane of tangency 104 (step 130) to arrive at a second proposed virtual tool 28 location.

If none of the tool 28 points would penetrate the virtual surface 25 at this second proposed virtual tool position (step 132), then the haptic rendering process 16 moves the virtual tool origin 27 to the second proposed location (step 121) and

returns to step 120. This situation represents "Case 2" and the haptic rendering process 16 has assumed that the virtual tool 28 is intended to be touching the surface 25 of the virtual object 26.

Case 3 of Tool Encountering Edge Condition

If at the second proposed tool position, some points of the virtual tool 28 would penetrate the virtual surface 25 (step 132), then the haptic rendering process 16 finds the two surface direction vectors 181a and 181b of the virtual object 26 at the tool's 28 two points of greatest penetration at each of the previous two proposed tool positions (step 136). FIG 8A illustrates a virtual tool 28 encountering the outside or convex edge 177 of a virtual object 26 in one embodiment of the invention. The virtual object 26 extends out of the plane of the diagram, and has an edge 177 that likewise extends out of the plane of the diagram. In FIG. 8B the proposed tool position penetrates the virtual object 26 at point $S4_1$ and the haptic rendering process 16 calculates a surface direction vector 181a relative to virtual surface 25a. In FIG. 8C, the haptic rendering process 16 proposes a second proposed tool location based on the surface direction vector 181a relative to virtual surface 25a, as shown in FIG. 8C. In this proposed tool location, point $S3_2$ has penetrated the virtual object 26, and the haptic rendering process 16 calculates the surface direction vector 181b relative to virtual surface 25b.

The haptic rendering process 16 uses the vectors 181a and 181b and the desired direction of movement to determine constraints to the tool motion, as follows. The haptic rendering process 16 calculates the cross product of the vectors 181a, 181b found in step 136 to generate an edge line and a direction of movement 182 (step 138 in FIG. 7B), as shown in FIG. 8D. The haptic rendering process 16 then determines the dot product of each minimum distance vector

181a, 181b with the direction of movement 182, as indicated by the user (step 140). The haptic rendering process 16 then determines if either dot product is greater than zero (step 140). If one of the dot products is greater than zero, then the haptic rendering process 16 takes the average of both surface direction vectors 181a, 181b (step 142). The haptic rendering process 16 then proposes to move the tool 28 in the direction indicated by the average vector (step 144) and then proceeds to Case 2 (step 146). The haptic rendering process 16 then proceeds to step 127 in FIG. 7A to determine a plane of tangency 104 using the average normal.

If neither dot product is greater than zero (step 140), then the haptic rendering process 16 constrains movement to the previously determined edge line (step 148). See FIG. 8D, which shows the virtual tool 28 at the edge 177 of the virtual object 26. In one embodiment, the virtual tool 28 is constrained based on one point of the tool 28, such as the origin 27 of the tool 28. The haptic rendering process 16 then attempts to move the virtual tool origin 27 toward the haptic interface location 98 but constrains the movement to the direction of the edge 108 or 177 (step 148). If none of the tool points would penetrate the virtual surface 25 at this third proposed virtual tool position (checked in step 149), then the haptic rendering process 16 moves the virtual tool origin 27 to this proposed location (step 121) and returns to step 120.

This situation represents "Case 3," and the haptic rendering process 16 assumes that the virtual tool 28 is on an edge 108 or 177 of the virtual object 26. Generally, in Case 3, the haptic rendering process 16 attempts to properly identify an edge 108 or 177 of the virtual object 26 and allow movement of the virtual tool 28 along the edge 108 or 177.

In one embodiment the haptic rendering process 16 identifies an edge 108 or boundary between two portions of the virtual surface 25 of a virtual object 26. The two portions are not required to be planar. For example, the two portions may have curved surfaces. The two portions are not required to be in contact but may represent portions of virtual surfaces 25 of two separate virtual objects 26 located in contact with or near each other.

Case 4 of Tool Encountering A Hole

If one or more points of the virtual tool penetrate the virtual surface 25 (step 149), the haptic rendering process 16 finds the surface direction vector 101 at the point of greatest potential penetration of the virtual object 26 by the virtual tool 28 (step 154). The haptic rendering process 16 then determines the penetration value at the point of greatest potential penetration and saves this value for subsequent use (step 156). The haptic rendering process 16 then attempts to move the virtual tool 28 in the direction of the surface direction vector 101 just determined (step 158). The haptic rendering process 16 then checks to see if no points of the virtual tool 28 at the proposed location penetrate the virtual surface 25 (step 160). If no points penetrate the virtual surface 25 86, then the haptic rendering process 16 moves the tool 28 to the proposed location (step 121) and proceeds to step 120. This situation represents "Case 4," which occurs if the virtual tool 28 is in a corner, recess, or "hole" in the virtual object 26.

If one or more points of the virtual tool 28 penetrate the virtual surface 25, then the haptic rendering process 16 proceeds to step 162 and determines if the penetration values at all the tool points would be less than the previously determined penetration value at the deepest potential penetration, as calculated in step 156. If all the tool points would be less than the previously determined penetration value, then the haptic rendering process 16 returns to step 154.

If all the tool points would not be less than the previously determined penetration value, then the haptic rendering process 16 proposes moving the virtual tool 28 in a random direction (step 166) from its previous potential position and goes to step 149.

Description of a Voxel-Based Approach

In one embodiment of the invention, the virtual object 26 is implemented as a volume using concepts of voxels 78, density, and an isosurface 86. FIG. 9 shows several voxels 78 having different density values 0, 51, 102, 153, 204, and 255, a gradient 80 established by these voxel values, and an isosurface 86. As used herein, density is not a physical property of the virtual object 26, but rather an abstract construct used for determining the shape of the virtual object 26. FIG. 10 illustrates a pictorial view in two dimensions of the voxels 78 of a three dimensional rectangular solid 82 with an isosurface 86 at value 128 indicating the solid surface of the rectangular solid 82. As shown in FIG. 10, the volume of the virtual object 26 is modeled using a 3-D array of evenly spaced elements or voxels 78 located at discrete points in the virtual environment 26. In another embodiment, the elements are not required to be arranged with regular spacing. Each voxel 78 stores a density value. Density values for points that lie between the discrete voxel points can also be evaluated using interpolation. The volume also stores a density threshold value. Points having a density value greater than the density threshold are considered to be inside the object. Points having a density value less than the density threshold are considered to be outside the object. As used herein, an "isosurface" 86 refers to a locus of points all having an identical density value. In one embodiment, the isosurface 86 whose density value equals the density threshold represents the virtual surface 25 of the virtual object 26. In one embodiment, as shown in FIG. 10 this density threshold is 128

and the voxel density values can range from 0 to 255. Thus a voxel representation facilitates an easy method for determining whether points on a virtual tool 28 lie inside or outside of a virtual surface 25 of a virtual object 26. All of the voxels 78 shown in FIG, 10 are associated with the rectangular solid 82, but a user moving a virtual tool 28 toward the rectangular solid 82 would not encounter a solid feeling surface until the virtual tool 28 contacts the isosurface 86. The user would not feel any resistance to moving the tool 28 when moving through the voxels 78 with density values, such 0, 51, and 102, which are less than 128.

IMPLEMENTATION OF THE HAPTIC RENDERING PROCESS

The haptic rendering process 16 between a virtual object 26 and virtual tool 28 is described in more detail below for one embodiment of the invention as implemented by the assignee, SensAble Technologies, Inc. of Cambridge, Massachusetts.

The chosen volumetric representation is integrated with the GHOST® SDK (Software Developer's Kit) haptic interaction software developed by SensAble Technologies, which provides much of the necessary haptics functionality and reduces haptic virtual objects into a set of basic methods that are then handled correctly by GHOST® SDK. The GHOST® SDK uses the c++ programming language. The developer can create a high-level object which needs only to handle basic interactions such as determining vectors, without being required to address low-level processes such as generating forces on the haptics device, resolving multiple collisions, and other more complex issues addressed by the GHOST software.

Volume Implementation Using Voxels

Haptic virtual objects are handled by a volume class. One embodiment of the invention is the *gstVolume* class. The *gstVolume* class follows the specifications of the generally provided *gstShape* GHOST class and follows the behavior of general geometric classes.

As described previously, the volume is represented using a voxel grid, the density values of which define an implicit virtual surface or isosurface 86 for the virtual object 26 as described for FIGS. 9 and 10. A valid volume for this representation is created containing an established gradient 80.

The specific voxel values defining the gradient 80 depend on the shape of virtual object 26 being presented. In one embodiment, the voxel values may vary between 0 and 255, with the value 128 representing the virtual surface. Any modifications to the volume must also preserve the gradient 80 to avoid incorrect calculation of surface direction vectors 101 or penetration distances.

The voxel value at any point gives an indication of the penetration depth and the shortest distance between that voxel 78 and the surface 86 of the volume. The ramp length is the number of voxels 78 over which density values go from their minimum (0) to their maximum (255). Voxel values increase with the penetration depth until the values reach a maximum. All voxels 78 beyond that penetration depth are also set to that maximum value. Likewise, when moving farther from the virtual surface in a direction away from the volume, voxel values decrease until they reach the minimum value. All values in that direction beyond that distance are set to the minimum value.

The volume class is used to determine approximate surface direction vectors 101 to the surface 86 from any internal point or point along the surface 86

of the volume using the density gradient 80 as explained below. FIG. 11 illustrates a ramp length diagram comparing voxel values to penetration distances for one embodiment of the invention. FIG. 11 depicts a vertical axis 198 for voxel values, a horizontal axis 202 for the penetration distance, an isosurface line 205 representing a density threshold value of 128, a ramp length 206, a maximum voxel value 208 with a value of 255, and a penetration line 210 at the isosurface 86 corresponding to the density threshold value 128. The penetration line 210 indicates that any penetration to the right of the line is into the solid body of the virtual object 26.

Ramp Length

The maximum penetration depth where a gradient 80 exists is defined by the ramp length 206, the density range, and the density threshold value. For example, for a ramp length 206 with a value of 4, a density range of 0 to 255 and a density threshold of 128, any penetration beyond 1.99 voxels [4/255*(255-128)] from the isosurface 86 will not lie within the portion of the volume where the voxel values exhibit a gradient. As such, the amount of penetration beyond this distance cannot be reliably calculated. Beyond that, the voxel values will all be at the maximum value. In one embodiment, the maximum value is 255.

In one embodiment, the direction of the surface direction vector 101 is calculated to be the direction of largest voxel density value gradient at that point. For any point that lies within the portion of the volume which exhibits a density gradient 80 a surface direction vector 101 can be calculated which points to the virtual surface and whose magnitude is the distance to the surface 86. In other words, in this region of space, the volume class can return a surface direction vector 101 which, when added to the current location, returns a location that is near the surface 86 of the volume. The vector calculations contain the same

limitations as the penetration distance calculations; that is, the surface direction vector 101 can only be calculated reliably within the portion of the volume which exhibits a density gradient 80.

Interpolating Values for Intermediate Points

Although the volume is characterized by a discrete array of voxels 78, it must be able to return a valid value at any point along the continuous range of its space, since the object it represents is continuous. For example, a line is characterized by two discrete points, but itself is continuous and can be evaluated at any point. If a value is requested at a point where a voxel 78 is present, then that particular voxel density value is returned. For spaces between voxels 78, the value may be resolved through tri-linear interpolation, a method of converting a discrete set of points into a continuous field. In another embodiment, the interpolation can be based on other methods, such as a quadric interpolation.

In one embodiment, the evaluation is accomplished in a single step via a mathematical formula that weights the contribution of each voxel 78 by its distance to the point being evaluated.

In one embodiment, voxels 78 are spaced one millimeter (that is, one world coordinate in a coordinate system based on millimeter spacing) apart. In other embodiments, the voxels 78 are spaced other distances apart. In general, there is constant spacing between each of the voxels 78, but there need not be for some embodiments. In further embodiments, memory management techniques may be utilized. A larger number of voxels 78 can be represented, and thus either the resolution or size of the object 26 can be increased.

Calculating Vectors

As stated previously, the required surface direction vector 101 at any point is determined using the direction of maximum density gradient at that point. This maximum density gradient is determined using central differences: the density value at a set distance from the point in the direction of each Cartesian coordinate in turn is determined, and the differences between those values determines the vector direction.

FIG. 12 illustrates a method of calculating the surface direction vector 101 of a point based on the density values of surrounding evaluation points. FIG. 12 shows a point 230 for which a direction vector 101 is being calculated. The surrounding evaluation points include a first evaluation point 232 with a value of 40, a second evaluation point 234 with a value of 100, a third evaluation point 236 with a value of 80, and a fourth evaluation point 238 with a value of 40. In one embodiment, each evaluation point is a voxel 78 and the values represent density values.

The vector and density computations are used to project any point within the virtual object 26 to the virtual surface 25 of the virtual object 26. Typically, this projection is used for the purposes of calculating the potential tool surface contact point 226. If a tool point 230 is penetrating deeper than the region in which a gradient exists 206, then no projection can be made directly from that point 230. Instead, the point must be first brought closer to the surface 25 (that is, to a region where a gradient 80 exists).

FIG. 13 illustrates a region of a volume where a gradient 80 for a virtual object 26 exists, including a last tool point 224, a final tool surface contact point 226, an initial proposed point 242, and a second proposed point 244 for one

embodiment of the invention. In FIG. 13, the haptic rendering process 16 proposes to move the tool location such that a point 224 on the tool 28 would move to the point represented by 242. However, point 242 is not located within the region where a valid gradient can be computed. Because the maximum surface direction vector 101 cannot be calculated for this point, the haptic rendering process 16 calculates a second proposed tool position that would result in the tool point 242 moving to the second proposed point 244. Point 244 is the approximate midpoint of a line 246 between point 242 and the last tool point 224. If the voxel density value at that point 244 is still the maximum value (255 in one embodiment), the haptic rendering process 16 can continue to average the point 244 with the last tool point 224, by using a binary algorithm to bring it progressively closer to the last tool point 224, until the midpoint 244 lies within the gradient 80. Once within the region where a gradient exists, the surface direction vector 101 to the virtual surface from the latest midpoint 244 can be computed for the purposes previously described.

In one embodiment, a binary search method is used to determine the intersection point between the endpoints of the segment. The process is repeated until either the points are within a tolerable error of the desired value, or a maximum number of iterations has been reached.

FIGS. 14A, 14B, and 14C illustrate an example of a binary search for locating the virtual surface 25 for a segment 249 that intersects the virtual surface 25 for one embodiment of the invention. In FIG. 14A, the haptic rendering process determines a midpoint 254 by averaging the locations of the endpoints 248 and 252. In FIG. 14B the haptic rendering process 16 treats the point 254 as an endpoint of a modified segment 253 extending from point 254 to endpoint 248. The haptic rendering process 16 determines a midpoint 256 between points 248

and 254 and how far points 248 and 254 are from the virtual surface 25. The haptic rendering process 16 determines that point 248 is farther from the virtual surface 25 than point 254, and thus sets the upper endpoint 248 to be at a new point 256. The endpoints of the additionally modified segment 257 are thus points 256 and 254. This process is repeated until a point is found that is regarded as the intersection point 258 (within a predetermined distance from the virtual surface 25) of the original segment 249 and the virtual surface 25, as shown in FIG. 14C.

OTHER INTERACTION TECHNIQUES

The following sections describe in more detail an implementation of virtual object 26 and virtual tool 28 interaction according to an embodiment of the invention implemented by the assignee, SensAble Technologies, Inc. of Cambridge, Massachusetts.

Single Point Methods

In some contexts a single point virtual tool 28 may interact with the virtual object 26. This may be handled as the simplest case of multipoint interaction using the methods previously described and described in more detail later. In another embodiment, specialized methods may be used for single point interaction, as described herein.

Collision Detection for Single Point Interaction

The haptic rendering algorithm determines when a collision between a virtual tool 28 and a virtual object 26 has occurred. A collision is detected when the haptic rendering process 16 attempts to move a tool 28 to penetrate a surface. In one embodiment, a collision occurs whenever the haptic interface location crosses through the virtual surface 25. In one embodiment, the virtual

surface 25 may be a NURBS surface. In one embodiment, the virtual surface 25 may be the "skin" of a volumetric solid.

A stateless haptic rendering algorithm would consider only the haptic interface location 98 in determining the resulting forces from a collision. It would not consider any history or previous collisions. The resulting forces in a stateless algorithm would use only the current haptic interface location 98 to determine the 1) depth of penetration and 2) direction to the nearest surface 25. The resulting force vector would be a vector toward the nearest surface whose magnitude is proportional to the penetration depth. In the case of a voxel embodiment, the penetration depth of the haptic interface device point is generally proportional to the voxel density at that point.

The direction of the force vector would be a vector that points from the haptic interface location 98 to the closet point on the surface. In the case of the voxel embodiment, this vector is simply the maximum voxel gradient (i.e. direction of greatest voxel density change) at the haptic interface location 98. In conclusion, a stateless haptic rendering algorithm would consider the location of the haptic interface device at each iteration, determine if it has crossed a virtual surface 25 or is embedded in a virtual object 26, then return a force vector whose direction is from the haptic interface device location 98 to the nearest point on the surface, and whose magnitude is proportional to the penetration distance.

A stateless algorithm handles only basic cases of tool 28 interaction and falls short of accurately representing some cases. The most notable case for which the stateless algorithm fails is for the case of thin objects. If a user begins pressing through a thin surface, at some point the nearest surface point to the haptic interface device location 98 will be on the other side of the thin object

(i.e. after the user has pressed more than halfway through), and thus the force vector will incorrectly push the user out the other side.

An improved algorithm keeps track of a virtual tool position at each iteration. Maintaining this virtual tool position is an efficient way to contain state information about the history of the user's path or trajectory. If the haptic interface device location 98 moves across a surface, the algorithm will attempt to move the virtual tool 28 toward the haptic interface device location 98, but never through the surface, as if the two points were connected by a spring. The resulting force sent to the haptic interface device 10 is proportional to the distance between the tool 28 and the haptic interface device location. In some embodiments, the force is also proportional to the difference in velocity or acceleration between the virtual tool 28 position and the haptic interface device location 98. The tool position on the virtual surface 25 is referred to herein as the surface contact point or SCP 226.

In one embodiment, the haptic rendering process 16 attempts to minimize the distance between the SCP 226 and the current haptic interface location 98, given that a path of decreasing distance exists between the last SCP 224 and the desired one. The connection between the SCP 226 and the haptic interface location 98 can be viewed as a spring. The haptic rendering process 16 processes the locations of the SCP 226 and haptic interface location 98 in iterative steps. At each iteration, the haptic rendering process 16 attempts to minimize the distance from the SCP 226 to the haptic interface location 98 if possible.

In one embodiment, the haptic rendering process 16 uses an algorithm for determining the SCP 226 based on a stepping method. For a given number of iterations, the algorithm determines a valid direction to move the SCP 226 which would yield a better solution (that is, decrease the distance between the SCP 226

and the current haptic interface location 98) and moves the point from the previous SCP 224 to a current SCP 226 in that direction. In one embodiment, valid directions are those which move along the virtual surface 25 of the virtual object 26. The haptic rendering algorithm should not allow the SCP 226 to penetrate the virtual surface 25 as it steps toward the haptic interface location 98.

FIG. 15 shows one method of determining a final SCP 226 used in one embodiment. FIG. 15 illustrates changing SCP positions 264a, 264b, 264c, tangency planes 260a, 260b, 260c, and resulting SCP 226. The resulting SCP 226 is a stationary position until the user makes a further movement of the virtual tool 28. In one embodiment, the changing SCP 264 is similar to the last or previous SCP 224. In one embodiment, the tangency planes 260a, 260b, 260c are similar to the tangency plane 104 described earlier. Using this process, the haptic rendering process 16 creates a tangency plane 260a for an existing SCP 264a using the surface direction vector 101a for the SCP 264a. The haptic rendering process 16 then moves the SCP 264a a fixed step (predetermined distance) on the virtual surface 25 to new SCP 264b, constraining the movement of the changing SCP 264b to the tangency plane 260b. The haptic rendering process 16 then determines a new tangency plane 262b and repeats the stepping process to move the SCP 264b to a changed SCP 264c. The haptic rendering process 16 then creates a new tangency plane 262c and repeats the process until the SCP 264c reaches a final position at SCP 226 above, or at the shortest distance from, the haptic interface location 98, as shown in FIG. 15. This method of movement is also termed a march by the SCP 264 across the virtual surface 25.

In one embodiment, the SCP 264 is constrained to move in the plane 260, which passes through the current SCP 264 and whose vector is the approximate direction of the normal to the virtual surface 25 at that point. In other words, the

SCP 264 is allowed to move in any direction perpendicular to the surface direction vector 101 at that point. In one embodiment, the haptic rendering process 16 determines which direction along that plane 260 which would yield the optimum solution (that is, the solution that decreases the distance between the SCP 264 and the haptic interface location 98 by the largest amount) and moves the SCP 264 in that direction. Since the shape of the surface 25 at the SCP 264 position may be curved, moving linearly in some direction may either cause the SCP 264 to penetrate the virtual surface 25 (if convex) or leave the virtual surface 25 (if concave) causing an error in the SCP 264 position. Therefore, the SCP 264 may be projected onto the surface at each step. In one embodiment, this error is minimal if the step size is small, so the haptic rendering process 16 only needs to project the SCP 264 onto the virtual surface 25 after the final position (final surface contact point 226) has been determined from the march. Another potential problem involves local minimums and maximums. Because the SCP 264 only moves in directions which draw it progressively closer to the haptic interface location 98, the SCP 264 may not be able to traverse local maximums (that is, small humps in the virtual surface 25) and may settle in local minimums (small dips in the virtual surface 25). In one embodiment, the solution is to allow the spring (a spring calculation connecting the SCP 264 and haptic interface location 98) to stretch a small and finite distance so that the SCP 264 march can overcome these local discrepancies. In one embodiment this problem of local maximums and minimums rarely becomes a serious problem, so the spring stretching approach need not be implemented.

In another embodiment, the haptic rendering process 16 implements an iterative stepping algorithm as follows: The haptic rendering algorithm creates a plane 260 passing through the current or changing SCP 264, whose surface

direction vector 101 is the calculated approximate normal to the virtual surface at the SCP 264. The haptic rendering algorithm projects the current haptic interface location 98 onto the nearest point on that plane 260a, 260b, 260c and creates a vector 262a, 262b, 262c from the SCP 264a, 264b, 264c to that point. This vector then becomes the desired direction for the march. The haptic rendering algorithm moves the SCP 264a, 264b, 264c a fixed step in the direction indicated by the vector 262a, 262b, 262c. The haptic rendering algorithm repeats these steps. Finally, the haptic rendering algorithm projects the SCP 264a, 264b, 264c onto the virtual surface 25 using the intersection technique described above. In an alternate embodiment, the haptic rendering algorithm uses the faster but less robust technique of projecting based on the surface direction vector 101 and density value at that point. For example, see FIGS. 17A-17E.

COLLISION DETECTION WITH THREE-DIMENSIONAL TOOLS

Collision with a point interface is sufficient for many interactions, but in one embodiment a more complete method of three-dimensional tool interaction is used. The virtual tool 28 is represented by a series of points along its surface, as discussed previously. At each iteration, the haptic rendering process 16 tests each of these points on the surface of the tool 28 to test for penetration of the tool 28 into the virtual surface 25 of the virtual object 26.

Improved Stepping Method

One embodiment of the haptic rendering process 16 maintains a tool position and iteratively moves it toward the current haptic interface location 98. The haptic rendering process 16 operates in a repeated loop of iterative processing steps. For each loop, the haptic rendering algorithm attempts to minimize its

distance to the haptic interface location 98 without violating the virtual surface 25. In other words, at each iteration, the spring connection between the tool 28 and the haptic interface location 98 attempts to contract, but stops if that action would cause any of points of the virtual tool 28 to penetrate the virtual surface 25.

During each iteration, the haptic rendering process 16 attempts to march the tool 28 toward the haptic interface location 98, checks for violations against the virtual object 26, and repeats this process a number of times. Since the haptic rendering process 16 evaluates for collision at a potential tool location which is a distance equivalent to the step size away from the previous tool location at each step, the step size is bound to a relatively small value so that the haptic rendering process 16 does not jump the tool 28 over any features such as a concave "hole" in the surface 25 of the virtual object 26. The step size should also be large enough so that the movement of the tool 28 can reasonably keep up with the movement of the haptic interface location 98 as the user moves the haptic interface device 10. If the haptic interface device 10 moves a considerable distance but the step size of the tool 28 is small, the user feels a drag as the haptic rendering process 16 attempts to move the tool 28 to the haptic interface location 98. In one embodiment, the step size for each iteration should be less than the minimum feature size that the user will want to feel. In one voxel embodiment, the step size is one half of the voxel grid spacing.

Moving Along a Face of a Surface

If the movement method is to always move directly toward the haptic interface location 98, the tool 28 would get "stuck" on the virtual surface 25 once it touched the virtual surface 25. Therefore, the haptic rendering process 16 attempts to move the tool 28 along the surface 25 in a manner that minimizes its spring distance (between the tool 28 and the haptic interface location 98) instead

of simply backing up to the previous position whenever the tool 28 violates the surface 25. The methods for achieving this are similar to those for determining the SCP 226 for a single point.

FIGS. 17A-17E show a virtual tool 28 encountering a virtual surface 25 and moving along the virtual surface 25 constrained by a plane 300. In FIGS. 17A and 17B, the tool 28 is not penetrating the virtual surface 25, so the haptic rendering process moves the tool 28 to follow the location of the haptic interface location 98. In FIG. 17C the potential tool position would result in penetration of the virtual surface 25 and thus violate the interaction constraints. The haptic rendering process 16 determines the surface direction vector 101 at the point of greatest potential penetration 304. In FIG. 17D the haptic rendering process 16 determines a plane 300 perpendicular to the surface direction vector 101 and passing through the last legitimate tool position in figure 17B. The haptic rendering process 16 constrains the virtual tool 28 to move only within the plane 300. The desired direction of motion 306 is determined by taking a vector 308 from a tool origin 27 to the haptic interface location 98 and projecting that vector 308 onto the plane 300. In FIG. 17E, the tool 28 moves a fixed distance in the direction determined in FIG. 17D. This manner of movement effectively allows the tool 28 to move along the virtual surface 25 without getting "stuck." Once the haptic rendering process 16 determines that the tool 28 is near the virtual surface 25 (that is, after finding a potential penetration), this method may be used for all further calculations until the tool 28 leaves the virtual surface 25 (that is, the potential tool points cease to penetrate the virtual surface 25, causing the tool 28 to no longer be in a touching state).

Thus, the haptic rendering process 16 can detect when the tool 28 is on a face of a virtual surface 25, and attempts to move it along the surface 25 instead of directly toward the haptic interface location 98 when this condition is detected.

Moving Along or Across an Edge

More complicated situations may occur where the haptic rendering process 16 is moving the virtual tool 28 along an edge 108 or 177 instead of a face of a virtual object 26. In this situation, the haptic rendering process 16 constrains the tool 28 to a line (the edge) instead of a plane (the face). When multiple potential penetration points on the surface of the tool 28 have differing surface direction vectors 101, then the haptic rendering process 16 assumes that the tool 28 is at an intersection of the two faces with differing surface direction vectors 101, and constrains the tool 28 to that virtual edge 108 accordingly. The virtual edge 108 is the cross product of the surface direction vectors 101; the haptic rendering process 16 then constrains the tool 28 to moving only in that direction or its negative. (See the discussion of edge constraints associated with FIGS. 7A-7C.)

In one embodiment, it is not always the case that the haptic rendering process 16 constrains a virtual tool 28 to moving along an edge 108 or 177 when one is detected. Otherwise, when attempting to slide across edges 177 that are convex (that is, mountains or "sharp" edges 177 in the virtual surface 25 of the virtual object 26), the tool 28 may get stuck. Thus, the haptic rendering process 16 should distinguish between when the tool 28 is attempting to slide along a concave edge 108 versus when it is attempting to cross a convex edge 177. This is determined by taking the dot product of the desired direction (that is, a vector from the tool origin 27 to the haptic interface location 98) with the surface direction vectors 101 of each face that forms the virtual edge 108 or 177. See FIG. 7B. A positive dot product indicates that the tool 28 is attempting to move

away from one of the virtual surfaces 25 instead of sliding along the edge 108 or 177. If either dot product is positive, the haptic rendering process 16 assumes that the tool 28 is moving across a convex edge 177 and does not constrain itself to the convex edge 177. When this case is detected, the haptic rendering process 16 "pushes the tool 28 away" slightly, moving in the direction of the vector which is the average of the two surface direction vectors 101.

The methods described above are sufficient in most cases; however, scenarios exist where the movement of the tool 28 may get "stuck" and the haptic rendering process 16 is unable to move the tool 28 along the virtual surface 25 effectively. To compensate for this, the haptic rendering process 16 tries moving the tool 28 tries away from the surface 25 (that is, in the direction of the surface direction vector 101) at small increments if it has been stuck for a considerable time (that is, the haptic rendering process 16 has unsuccessfully attempted to move the tool 28 for the past several iterations).

Progressive Extraction

Finally, there are situations in which the tool 28 somehow becomes stuck inside the virtual surface 25. In other words, the current tool position is penetrating the surface 25. For example, if the user touches a surface 25 with the flat face of a cube, then rotates the cube such that one of the edges is now penetrating the surface 25. Unless the haptic device limits this rotation via a torque feedback, the haptic rendering process 16 can put the virtual tool 28 in a position of penetration, which violates the desired behavior. In one embodiment, the haptic interface device 10 would have six degrees of freedom of force feedback and thus not allow invalid rotations. In other embodiments, the haptic interface device 10 has more than six degrees of freedom. An embedded tool position may also occur if the user uses a modification mode to add material on

top of the tool 28, or the virtual object 26 is rotated such that the tool 28 suddenly becomes embedded.

If the tool 28 is forced to an invalid position, the haptic rendering process 16 needs some way of extracting the tool 28 to a valid position. In these cases, the haptic rendering process 16 should attempt to move the tool 28 in a direction away from the virtual surface 25 to escape the surface 25. Therefore, if the tool 28 is stuck such that moving directly toward the haptic interface location 98, moving along the constraint plane 300, moving to its previous position, and moving in the direction in the normal of that plane 300 all would result in penetration of the virtual object 26, then the haptic rendering process 16 attempts to "jump" the tool 28 a significant distance in the direction of the surface direction vector 101. (See FIG. 7C.) If this jump still does not free the tool 28 (that is, one of its points would remain embedded in the surface 25), but the proposed position results in lesser penetration of the virtual object 26, then it is considered a superior position (that is, if the point of greatest penetration yields a lower density evaluation value than the previous evaluation for the previous point of greatest penetration). This condition allows the haptic rendering process 16 to have a notion of "progress;" that is, even though moving in some direction does not fully release the tool 28 from the surface 25, the movement is still an improvement over the previous position if moving in this direction causes the penetration distance to decrease.

Random Extraction

The above methods handle most cases. However, some scenarios exist where the tool 28 is "stuck" but moving in the direction of the normal to the plane does not yield any improvement. This is a rare condition since the surface direction vector 101 points toward the surface 25 of the virtual object 26, moving in that direction should decrease the penetration distance. But although moving in

that direction decreases the penetration distance of that point which is used to determine the surface direction vector 101, this movement may increase the penetration of another point (usually one of the opposite side) such that the tool's 28 overall penetration distance increases. In circumstances where none of the above techniques yields acceptable results, the haptic rendering process 16 may conclude that the tool 28 is truly stuck and should not legally move (for example, as in FIGS. 16A and 16B).

For example, suppose a spherical tool 28 ends up in a situation where it is in a gap or channel 270 between two virtual objects 272 and 274, whose width is less than that of the tool 28. A similar situation occurs if the tool 28 is in a tunnel or conduit in a virtual object 26. FIGS. 16A and 16B show a spherical virtual tool 28 in a channel 270 formed between two virtual objects 272 and 274. The diameter of the channel 270 is less than the diameter of the virtual tool 28. FIG. 16A shows a point of greatest penetration 276 and a vector 278 to the surface 280 of virtual object 274. FIG. 16B shows a new point with greatest penetration 282 and its computed vector 284. The point 288 represents the tool surface contact point for the previous penetration of virtual object 274. The haptic rendering process 16 may determine that the point 276 with the greatest X value has the greatest penetration (based on a horizontal X axis for FIGS. 16A and 16B). It will then push the sphere 28 toward the minus X direction by some distance. During the next iteration, the point of greatest penetration will probably be the point 282 with the least X value. This will then cause the sphere 28 to move back in the X direction by some distance, etc. In this example, the haptic rendering process 16 could cause the tool 28 to oscillate indefinitely between the positions depicted in FIGS. 16A and 16B.

Thus, as a final method, the haptic rendering process 16 may attempt to move the tool 28 in a random direction as a way of arriving at a better solution (that is, one that decreases the greatest penetration distance). (See FIG. 7C.) However, in one embodiment, instead of moving in a purely random direction, the haptic rendering process 16 determines which of the points of the tool 28 have computed surface direction vectors 101 (that is are either embedded in the surface 25 or within the field of decreasing non-zero values surrounding the surface 25) and chooses one of those surface direction vectors 101 randomly as the direction to move in. The movement of the tool 28 obeys the same rules as described for other cases in FIGS. 7A-7C. If the maximum penetration distance does not decrease as a result of the attempted move, then the haptic rendering process 16 does not move the tool 28 but instead tries a different direction.

Note that this final method is fairly expensive computationally. Thus, when the tool 28 reaches this state, the haptic rendering process 16 may determine all the current tool penetration points only once, by creating a list of indices of points whose penetration values are within the virtual surface 25. In one embodiment using voxels 78, the points include those points with densities greater than the density of the threshold value (isosurface 86). For all successive iterations, the haptic rendering process 16 randomly chooses one of the points and moves the tool 28 in the direction of that point's surface direction vector 101.

Variable Step Size and Other Optimizations

Adaptive Step Size

In one embodiment, the user may feel a drag force if the haptic rendering process 16 is not able to maintain the movement of the tool 28 with the movement of the haptic interface device 10. For example, if the step size or

number of iterations is small, then the haptic rendering process 16 may take a considerable amount of time to move the tool 28 to its final position if the haptic interface device 10 has moved a significant distance. Thus, it is desirable to have an adaptive algorithm that interactively sets the step size depending on the movement of the haptic interface device 10. If the haptic interface device 10 is moving quickly, then the step size increases to compensate. If the haptic interface device 10 is not moving quickly, the step size is decreased so that the tool 28 can be placed with greater accuracy. A smaller step size also helps prevent undesired buzzing or vibration of the haptic interface device 10. Buzzing can occur when the haptic rendering process 16 cycles the tool 28 between multiple positions, attempting to settle but never finding rest. See FIGS. 16A and 16B. However, if the step size is decreased, these oscillations become very small and barely noticeable or do not occur at all. On the other hand, if the user is moving a large distance, then oscillations and "buzzing" do not occur because the haptic rendering process 16 is attempting to maintain the position of the tool 28 with the haptic interface location 98 of the haptic interface device 10 instead of having the tool 28 stay in one position. Thus, if during a given loop, the haptic interface device 10 is not moving rapidly, the step size of the tool 28 is decreased.

Adaptive Number of Iterations

If the computer processor is not performing any other process-consuming tasks, then the number of iterations per second can be increased safely. The most intensive operations occur if the user is not only interacting with the virtual object 26 but is modifying it in some manner. For example, if the haptic rendering process 16 is using the tool 28 to remove or add material, the calculations can consume much of the processing time. If the virtual object 26 is not being modified, the haptic rendering process 16 assumes it can use a greater portion of

the processing time and thus increases its number of iterations. In one embodiment, the number of iterations is increased by a factor of two. In other embodiments, other factors are used to determine the number of iterations.

Tool Point Collision Detection Optimization

The processing requirement per step can also be decreased if not all points along the tool 28 are evaluated at each loop. Evaluating a large number of points to check for penetration of the virtual surface 25 by the virtual tool 28 consumes a large amount of processing time. In one embodiment, a faster method attempts to optimize this by choosing only a set number of points to evaluate during each iteration, depending on which points are closest to the surface 25. In a voxel embodiment, during the first step in each loop, the haptic rendering process 16 evaluates the density values for the voxels 78 at all tool points and remembers the five points that evaluated to the greatest densities. For subsequent steps during that loop, it characterizes the tool 28 by those five points and thus does not perform any collision detection with any of the other tool points. This gives a significant performance increase. In one embodiment, the average tool 28 may have around 80 tool points, so evaluating the only 5 of 80 provides a large advantage. When this optimization is used, the number of iterations per loop is doubled to four, so that the haptic rendering process 16 only performs collision detection against every tool point every fourth step.

This can lead to situations where one of the tool points would violate the virtual surface 26, if that point was not one of the original five chosen during the first step. The haptic rendering process 16 however, does not usually place the tool 28 in situations where a valid position cannot be found following an evaluation of a potentially illegal series of steps.

Adaptive Stepping Behavior

In one embodiment if the haptic rendering process 16 has been moving the tool 28 toward the haptic interface location 98 without encountering intersections with virtual objects 26) for several iterations, then the haptic rendering process 16 assumes that the tool 28 is in empty space. For future iterations, instead of stepping toward the haptic interface location 98, the tool origin 27 is set to be coincident with the haptic interface location 98 at each step. Thus, the user does not feel any unwanted force effects. This continues until the tool 28 intersects with a virtual object 26, in which case the haptic rendering process 16 uses the previously discussed stepping algorithms to increment the tool position to a valid location. The number of legal steps needed before the tool 28 is set to the haptic interface location 98 may be defined to be one hundred iterations. In other embodiments, the number of legal steps may be less than or greater than one hundred iterations.

Optimize Transform Computations

For each collision detection evaluation between a tool point and the virtual surface 25, the point is transformed from the local coordinates of the tool 28 to the local coordinates of the virtual object 26. The greatest process consumption involves multiplying the point by the appropriate transformation matrix to account for the rotation of the tool 28 and virtual object 26. In one embodiment, instead of calculating this transformation for each loop, the points are only updated to the current orientation every one hundred iterations. When the points are updated, they are stored in a separate point array of local points, which is used to represent the tool 28 until it is again updated. Note that only the rotation of the points is updated every one hundred iterations; the translation of the points is accurately updated every loop to accurately reflect the potential position of the tool 28.

Additional Conditions Handling

Normally, the haptic rendering process 16 places the tool 28 to remain on the outside of any virtual object 26 with which it is interacting. If some small portion of the tool 28 becomes buried in the virtual object 26, the tool 28 can usually be extracted to the surface 86 using one or more of the techniques described above. However, situations exist where the tool 28 may instantaneously become completely buried inside a virtual object 26. For example, if the user adds a large amount of material over the current tool position, or the user begins the program with the tool position inside a virtual object 26, then the tool 28 is buried. If the tool 28 is ever inside a virtual object 26, the desired behavior is for the haptic rendering process 16 to gradually push the tool 28 away from the virtual object 26 in the direction of the nearest surface 86. In another embodiment, the tool 28 reaches a state where it is completely or nearly buried in a virtual object 26, then no forces are generated until the haptic rendering process 16 moves the tool 28 to exit completely the virtual object 26. It sets its position to the haptic interface location 98 for subsequent iterations and generates no collisions until the tool 28 reaches a position where every tool point is outside of the virtual object 26. In a voxel embodiment, the tool 28 is considered buried if its center evaluates to a density value approximate to the maximum density value (which is a value of 255 in one embodiment).

Tool Behavior Under Object Transformations

The tool position is stored in the local coordinates of the virtual object 26. Thus, if the virtual object 26 instantaneously changes scale, translation, or rotation, the tool position may jump, and the user feels a sudden "kick back" in the haptic interface device 10. To prevent this in one embodiment, a check is

made during the first potential step of the tool 28 to determine if the transform of the object 26 has changed significantly since the previous loop. If this is true, then the tool position is set to the haptic interface location 98, and therefore no force generated. For example, if an object 28 were scaled upward, the haptic rendering process 16 would bury the tool 28 in the virtual object 26 instead of kicking the tool 28 to the surface 86.

Tool Position Error

The tool 28 is most often not directly on the virtual surface 25. As discussed previously, the haptic rendering process 16 only moves the tool 28 in discrete steps and its movement is constrained from violating the surface 25. Thus, the tool 28 rarely settles directly on the surface 25. This difference in position is not particularly noticeable when the user is feeling the surface 25 with the virtual tool 28. Realistic feel does not depend on having the tool 28 always directly on the virtual surface 25. In one voxel embodiment, step sizes are generally around 0.1 grid space unit; thus, the user at most experiences an error of 0.1 grid space units from the surface 86. This is below the threshold where it would make a difference in the force felt by the user. In other embodiments, the step size may have other values than the 0.1 grid space unit.

GLOSSARY

Constraint

An imposed limitation on the motion of the virtual tool 28.

Density

A scalar property of each single voxel 78, used for defining the shape of a virtual object 26 in one embodiment.

Density Threshold Value

In one embodiment, the density value which defines the isosurface 86 which represents the virtual surface 25. Voxels 78 with densities above this value represent interior points on the virtual object 26. Voxels 78 with densities below this value represent points outside the virtual object 26.

Edge

A temporary geometric construct representing a potentially viable move direction when the virtual tool 28 is contacting the virtual object 26 in more than one place.

Gradient

The rate of change of the density values of the voxels 78 along a given vector in one embodiment.

Haptic Interface Location

The location in the virtual environment which corresponds to the key position of the haptic interface device 10 in the real-world space.

Haptic Rendering Process

The process responsible for generating objects 26 in the haptic virtual environment and producing high-fidelity force-feedback to the user in real world space.

Haptic Virtual Environment

A computer representation of a space where a user can interact with virtual objects 26 through the sense of touch.

Interaction Modes

Settings which alter the interaction between the virtual tool 28 and the virtual object 26.

Isosurface

A theoretical surface 86 defined by the locus of identical voxel density values.

Panic Escape

A method of extracting the virtual tool 28 from a trapped condition.

Ramp length

In one embodiment, the number of voxels 78 over which density values go from their minimum (0) to their maximum (255).

Real World Space

The true three-dimensional physical world in which people live and interact.

Surface Direction Vector

A vector 101 evaluated at a point in relation to a virtual surface 25. If the point is at the virtual surface 25, this vector 101 is the normal to the surface 25 at that point. If the point is not on the surface 25, the vector 101 represents the approximate shortest distance to the surface 25 and is also the approximate surface normal at the point where the vector 101 would intersect the surface 25.

Tool Surface Contact Point

A location on the virtual surface 25 where the virtual tool 28 is in contact with the virtual surface 25.

Trilinear interpolation

In one embodiment, a technique for interpolating the densities of nearby voxels 78 to derive the density at a location that lies between the discrete voxel locations.

Virtual Object Modification Process

The process responsible for making changes to the virtual object 26.

Virtual Object

A computer representation of an object.

Virtual Surface

A computer representation of the "skin" of a virtual object 26.

Virtual Tool

A computer representation of a tool which the user uses to interact with the virtual environment.

Virtual Tool Origin

The location on the virtual tool 28 which strives to be coincident with the haptic interface location 98.

Voxels

A set of locations in the virtual environment, each storing information used in defining the shape of a virtual object 26 in one embodiment.

Having described the preferred embodiments of the invention, it will now become apparent to one of skill in the art that other embodiments incorporating the concepts may be used. It is felt, therefore, that the invention should not be limited to disclosed embodiments, but rather should be limited only by the spirit and scope of the following claims.